Package 'ggsem'

May 26, 2026

Type Package

Title Interactive Structural Equation Modeling (SEM) and Multi-Group Path Diagrams

Version 1.0.0

Maintainer Seung Hyun Min <seung.min@mail.mcgill.ca>

Description Provides an interactive workflow for visualizing structural equation modeling (SEM), multi-group path diagrams, and network diagrams in R. Users can directly manipulate nodes and edges to create publication-quality figures while maintaining statistical model integrity. Supports integration with 'lavaan', 'OpenMx', 'tidySEM', and 'blavaan' etc. Features include parameter-based aesthetic mapping, generative AI assistance, and complete reproducibility by exporting metadata for script-based workflows.

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports blavaan, DiagrammeRsvg, dplyr, ggplot2 (>= 4.0.0), grDevices, igraph, lavaan (>= 0.6.21), methods, network, purrr, qgraph, RColorBrewer, rlang, Rtsne, semPlot (>= 1.1.7), smplot2, stringr, tidyr, tidySEM, umap, xml2

Suggests colourpicker, DT, DiagrammeR, grid, memoise, shiny, shinyjs, svglite

URL https://smin95.github.io/ggsem/

BugReports https://github.com/smin95/ggsem/issues/

Config/Needs/website rmarkdown

NeedsCompilation no

Author Seung Hyun Min [aut, cre]

Repository CRAN

Date/Publication 2026-05-26 09:00:02 UTC

Contents

`add_group` *Add a SEM group to the visualization*

Description

Add a SEM group to the visualization

Usage

```
add_group(
  builder,
  name,
  model = NULL,
  object = NULL,
  level = NULL,
  x = 0,
  y = 0,
  width = NULL,
  height = NULL,
  type = NULL
)
```

Arguments

builder	A ggsem_builder object
name	Group name (required)
model	Model object (lavaan, blavaan, etc.)
object	Visualization object (qgraph, semPaths, etc.)
level	Group level of multi-group model or multi-group data (e.g., 'Pasteur')
x	X-coordinate for center position (default: 0)
y	Y-coordinate for center position (default: 0)
width	Width of visualization area (default: 25)
height	Height of visualization area (default: 25)
type	Type of analysis: 'sem' or 'network' (auto-detected)

Value

Updated ggsem_builder object

---

adjust_axis_range *Adjust Axis Range of a Plot of a ggplot2 Plot*

Description

This function modifies the axis ranges of a ggplot object, with optional user-specified ranges, additional buffers, and the ability to enforce a fixed aspect ratio. This is a modified version of `adjust_axis_space()`.

Usage

```
adjust_axis_range(
  plot,
  x_range = NULL,
  y_range = NULL,
  buffer_percent = 0,
  fixed_aspect_ratio = TRUE
)
```

Arguments

plot	A ggplot object. The plot whose axis ranges are to be adjusted.
x_range	A numeric vector of length 2 specifying the desired x-axis range. If `NULL`, the current x-axis range is retained. Default is `NULL`.
y_range	A numeric vector of length 2 specifying the desired y-axis range. If `NULL`, the current y-axis range is retained. Default is `NULL`.

`buffer_percent` A numeric value indicating the percentage of additional space to add to each axis range as a buffer. Default is ‘0‘ (no buffer).

`fixed_aspect_ratio`
A logical value indicating whether to maintain a fixed aspect ratio for the plot. If `TRUE`, the function adjusts one axis to preserve the aspect ratio. Default is `TRUE`.

Details

- If ‘x_range‘ or ‘y_range‘ are provided, these values will override the current axis ranges. - The ‘buffer_percent‘ parameter adds proportional space to the axis ranges, calculated as a percentage of the range’s width or height. - When ‘fixed_aspect_ratio‘ is ‘TRUE‘, the function adjusts either the x-axis or y-axis to ensure the plot maintains a fixed aspect ratio.

Value

A modified ggplot object with adjusted axis ranges.

Examples

```
# CSV files from ggsem app
points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1,
width_height_ratio = 1.6, orientation = 45, lavaan = FALSE,
network = FALSE, locked = FALSE, group = 1
)

lines_data <- data.frame(
x_start = 2, y_start = -2, x_end = 10, y_end = -2, ctrl_x = NA, ctrl_y = NA,
ctrl_x2 = NA, ctrl_y2 = NA, curvature_magnitude = NA, rotate_curvature = NA,
curvature_asymmetry = NA, type = 'Straight Line', color = '#000000',
end_color = NA, color_type = 'Single',
gradient_position = NA, width = 1.5, alpha = 1, arrow = FALSE,
arrow_type = NA, arrow_size = NA, two_way = FALSE, lavaan = FALSE,
network = FALSE, line_style = 'solid', locked = FALSE, group = 1
)

p <- csv_to_ggplot(graphics_data = list(points_data, lines_data),
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points


adjust_axis_range(p, x_range = c(-30,30), y_range= c(-30,30))
```

adjust_axis_space *Adjust Surrounding White Space of a ggplot2 Plot*

Description

This function allows users to remove or manage whitespace around graphical elements. It supports asymmetrical adjustments for each boundary (left, right, bottom, and top). Users can also maintain a fixed aspect ratio if required.

Usage

```
adjust_axis_space(
  plot,
  x_adjust_left_percent = 0,
  x_adjust_right_percent = 0,
  y_adjust_bottom_percent = 0,
  y_adjust_top_percent = 0,
  fixed_aspect_ratio = TRUE
)
```

Arguments

`plot`	A ggplot2 object. The plot whose axis ranges need adjustment.
`x_adjust_left_percent`	Numeric. Percentage by which to expand the left boundary of the x-axis. Default is `0`.
`x_adjust_right_percent`	Numeric. Percentage by which to expand the right boundary of the x-axis. Default is `0`.
`y_adjust_bottom_percent`	Numeric. Percentage by which to expand the bottom boundary of the y-axis. Default is `0`.
`y_adjust_top_percent`	Numeric. Percentage by which to expand the top boundary of the y-axis. Default is `0`‘.
`fixed_aspect_ratio`	Logical. If `TRUE`, maintains a fixed aspect ratio (1:1). If ‘FALSE‘, allows independent scaling for x and y axes. Default is `TRUE`.

Details

- **Percentage Adjustments:** The percentages provided for each axis boundary are calculated based on the current axis range. For example, `x_adjust_left_percent = 10` expands the left boundary by 10 - **Fixed Aspect Ratio:** When `fixed_aspect_ratio = TRUE`, the function adjusts either the x-axis or y-axis to maintain a 1:1 aspect ratio. The larger adjustment determines the scaling for both axes.

Value

A ggplot2 object with adjusted axis ranges. The adjusted plot retains its original attributes and is compatible with additional ggplot2 layers and themes.

Examples

```
# CSV files from ggsem app
points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1,
width_height_ratio = 1.6, orientation = 45, lavaan = FALSE,
network = FALSE, locked = FALSE, group = 1
)

lines_data <- data.frame(
x_start = 2, y_start = -2, x_end = 10, y_end = -2, ctrl_x = NA, ctrl_y = NA,
ctrl_x2 = NA, ctrl_y2 = NA, curvature_magnitude = NA, rotate_curvature = NA,
curvature_asymmetry = NA, type = 'Straight Line', color = '#000000',
end_color = NA, color_type = 'Single',
gradient_position = NA, width = 1.5, alpha = 1, arrow = FALSE,
arrow_type = NA, arrow_size = NA, two_way = FALSE, lavaan = FALSE,
network = FALSE, line_style = 'solid', locked = FALSE, group = 1
)


p <- csv_to_ggplot(graphics_data = list(points_data, lines_data),
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points


adjust_axis_space(p, x_adjust_left_percent = 10, x_adjust_right_percent = 10,
             y_adjust_bottom_percent = 5, y_adjust_top_percent = 5)
```

`csv_to_ggplot` *Convert CSV Files (from ggsem Shiny App) to a ggplot Object*

Description

This function converts the CSV files exported from the ggsem Shiny app into a customizable ggplot object. The resulting plot is compatible with ggplot2 functions, allowing users to modify it further (e.g., adding titles or annotations).

Usage

```
csv_to_ggplot(
  graphics_data = NULL,
  element_order = c("lines", "points", "loops", "annotations"),
  zoom_level = 1,
  horizontal_position = 0,
  vertical_position = 0,
  n = 100
)
```

Arguments

graphics_data	A list of data frames containing point data, line data, annotation data, and loop data. It is exported from the ggsem Shiny app. Default is NULL.
element_order	A character vector specifying the order in which graphical elements are added to the plot. For example: c("annotations", "loops", "lines", "points"). Elements at the front appear on top. Default includes all elements.
zoom_level	A numeric value controlling the zoom level of the plot. A value >1 zooms in; <1 zooms out. Default is 1.
horizontal_position	A numeric value to shift the plot horizontally. Default is 0.
vertical_position	A numeric value to shift the plot vertically. Default is 0.
n	Number of points used for interpolation in gradient or curved lines. Default is 100.

Details

- The function uses ‘coord_fixed‘ to ensure square plotting space and uniform scaling. - The ‘element_order‘ parameter determines the layering of graphical elements, with later elements appearing on top. - The ‘axis_ranges‘ attribute is attached to the plot for additional programmatic access.

Value

A ggplot object with an axis_ranges attribute specifying the x and y axis ranges after adjustments.

Examples

```
# CSV files from ggsem app
points_data <- data.frame(
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1,
width_height_ratio = 1.6, orientation = 45, lavaan = FALSE,
network = FALSE, locked = FALSE, group = 1
)

lines_data <- data.frame(
x_start = 2, y_start = -2, x_end = 10, y_end = -2, ctrl_x = NA, ctrl_y = NA,
ctrl_x2 = NA, ctrl_y2 = NA, curvature_magnitude = NA, rotate_curvature = NA,
```

```
curvature_asymmetry = NA, type = 'Straight Line', color = '#000000',
end_color = NA, color_type = 'Single',
gradient_position = NA, width = 1.5, alpha = 1, arrow = FALSE,
arrow_type = NA, arrow_size = NA, two_way = FALSE, lavaan = FALSE,
network = FALSE, line_style = 'solid', locked = FALSE, group = 1
)

csv_to_ggplot(graphics_data = list(points_data, lines_data),
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points
```

`draw_annotations` *Draw Text Annotations to a ggplot Object*

Description

This function overlays text annotations onto any ggplot object. It is particularly useful for adding annotations from CSV files generated by the ggsem Shiny app but can also be used with custom annotation data.

Usage

```
draw_annotations(annotations_data, zoom_level = 1)
```

Arguments

`annotations_data`

A data frame containing annotation information. Typically, this comes from a CSV file generated by the ggsem Shiny app. The required columns include:

- `text`: The text to annotate (character).
- `x`, `y`: The coordinates for the text (numeric).
- `font`: The font family to use (character, e.g., "serif").
- `size`: The size of the text (numeric).
- `color`: The color of the text (character, valid hex color).
- `angle`: The rotation angle of the text (numeric, in degrees).
- `alpha`: The transparency of the text (numeric, 0 to 1).
- `fontface`: The font style (character, e.g., "bold").
- `math_expression`: Logical, whether the text should be parsed as a mathematical expression.

`zoom_level` Numeric. Adjusts the size of annotations based on the zoom level. Default is `1`.

Value

ggplot2 annotation layers

Examples

```
library(ggplot2)

annotations_data <- data.frame(
text = 'x1', x = 9.5, y = 21, font = 'sans',
size = 16, color = '#FFFFFF', fill = NA, angle = 0,
alpha = 1, fontface = 'plain', math_expression = FALSE,
lavaan = FALSE, network = FALSE, locked = FALSE,
group_label = FALSE, loop_label = FALSE, group = 1
)

p <- ggplot()

p + draw_annotations(annotations_data, zoom_level = 1.2)
```

`draw_lines` *Draw Lines on a ggplot Object from Line Data*

Description

This function overlays lines or arrows to a ggplot object based on line data. It supports straight lines, curved lines, gradient color transitions, and one-way or two-way arrows. The data can come from a CSV file generated by the ggsem Shiny app or custom input.

Usage

```
draw_lines(lines_data, zoom_level = 1, n = 100)
```

Arguments

`lines_data` A data frame containing line information. The expected columns include:

- `x_start`, `y_start`: Starting coordinates of the line.
- `x_end`, `y_end`: Ending coordinates of the line.
- `ctrl_x`, `ctrl_y`: Control points for curved lines (optional).
- `type`: Type of line (`"Straight Line"`, `"Curved Line"`, `"Straight Arrow"`, or `"Curved Arrow"`).
- `color`: Start color of the line (hexadecimal color code).
- `end_color`: End color of the line for gradients (hexadecimal color code).
- `color_type`: `"Gradient"` for gradient lines, or `"Single"` for solid-colored lines.
- `gradient_position`: Position of the gradient transition along the line (numeric, 0 to 1).
- `width`: Width of the line (numeric).
- `alpha`: Transparency of the line (numeric, 0 to 1).
- `arrow`: Logical, whether an arrowhead is used.
- `arrow_type`: Type of arrow (`"open"`, `"closed"`, etc.).

- `arrow_size`: Size of the arrowhead.
- `two_way`: Logical, whether the line has two arrowheads (bidirectional).
- `line_style`: Line style (`"solid"`, `"dashed"`, or `"dotted"`).

`zoom_level` Numeric. Adjusts the size of line widths and arrowheads relative to the plot. Default is `1`.

`n` Integer. Number of points for interpolation in gradient or curved lines. Default is `100`.

Value

ggplot2 line layers

Examples

```
library(ggplot2)

lines_df <- data.frame(
x_start = 11, y_start = -2.3, x_end = 21, y_end = 3.5,
ctrl_x = NA, ctrl_y = NA, ctrl_x2 = NA, ctrl_y2 = NA,
curvature_magnitude = NA, rotate_curvature = NA,
curvature_asymmetry = NA, type = 'Straight Line',
color = '#000000', end_color = NA, color_type = 'Single',
gradient_position = NA, width = 1, alpha = 1, arrow = TRUE,
arrow_type = 'closed', arrow_size = 0.1, two_way = FALSE,
lavaan = FALSE, network = FALSE, line_style = 'solid',
locked = FALSE, group = 1
)

p <- ggplot()

p + draw_lines(lines_data = lines_df, zoom_level = 1.2, n = 200)
```

`draw_loops` *Draw Self-loop Arrows on a ggplot Object*

Description

This function overlays self-loop arrows to a ggplot object based on data describing their positions, sizes, orientations, and styles. Self-loop arrows can be drawn in one direction or bidirectionally with customizable parameters such as color, width, and arrow type. The data can come from a CSV file generated by the ggsem Shiny app or custom input.

Usage

```
draw_loops(loops_data, zoom_level = 1)
```

Arguments

loops_data	A data frame containing information about the self-loop arrows. The expected columns include: • x_center, y_center: Center coordinates of the loop. • radius: Radius of the loop. • color: Color of the loop (hexadecimal color code). • width: Width of the loop line (numeric). • alpha: Transparency of the loop line (numeric, 0 to 1). • arrow_type: Type of arrow ("closed" or "open"). • arrow_size: Size of the arrowhead. • gap_size: Size of the gap in the loop, specified as a fraction of the full circle (numeric, 0 to 1). • loop_width, loop_height: Width and height scaling factors for the loop. • orientation: Rotation angle of the loop in degrees. • two_way: Logical, whether the loop is bidirectional (adds arrows at both ends).
zoom_level	Numeric. Adjusts the size of line widths and arrowheads relative to the plot. Default is 1.

Value

ggplot2 loop layers

Examples

```
library(ggplot2)

loops_data <- data.frame(
x_center = -5, y_center = 5, radius = 2, color = '#000000',
width = 1, alpha = 1, arrow_type = 'closed', arrow_size = 0.1,
gap_size = 0.2, loop_width = 5, loop_height = 5, orientation = 0,
lavaan = TRUE, two_way = FALSE, locked = FALSE, group = 1
)

p <- ggplot()

p + draw_loops(loops_data, zoom_level = 1.2)
```

---

draw_points *Draw Points on a ggplot Object*

Description

This function overlays points to a ggplot object using data from a CSV file generated by the ggsem Shiny app or any custom dataset. Points can be styled with various shapes, colors, sizes, and orientations.

Usage

```
draw_points(points_data, zoom_level = 1)
```

Arguments

points_data	A data frame containing information about the points to be drawn. The expected columns include: • x, y: Coordinates of the point. • shape: Shape of the point ("circle", "square", "triangle", "rectangle", "oval", or "diamond"). • color: Fill color of the point (hexadecimal color code). • size: Size of the point. • border_color: Border color of the point (hexadecimal color code). • border_width: Width of the border. • alpha: Transparency of the point (numeric, 0 to 1). • width_height_ratio: Ratio of width to height (for shapes like rectangles and ovals). • orientation: Rotation angle of the point in degrees (for shapes like rectangles and diamonds).
zoom_level	Numeric. Adjusts the size of the points relative to the plot. Default is 1.

Value

ggplot2 point layers

Examples

```
library(ggplot2)

points_data <- data.frame(
x = 0, y = 0, shape = 'square', color = '#1262B3', size = 12,
border_color = '#FFFFFF', border_width = 1, alpha = 1,
width_height_ratio = 1.6, orientation = 0,
lavaan = FALSE, network = FALSE, locked = FALSE,
group = 1
)

p <- ggplot()

p + draw_points(points_data, zoom_level = 1.2) +
scale_x_continuous(limits = c(0,50)) +
scale_y_continuous(limits = c(0,50))
```

get_axis_range *Get axis range of a ggplot object*

Description

A function to calculate the range of x- and y- axes.

Usage

```
get_axis_range(plot)
```

Arguments

plot	ggplot output from csv_to_ggplot()

Value

A list object that has two elements, each of which has two vector values. The first element stores the minimum and maximum values of the plot's x-axis range, while the second element stores the minimum and maximum values of the plot's y-axis range.

Examples

```
library(ggplot2)
ggplot(mtcars) + geom_point(aes(mpg, disp)) -> p1
get_axis_range(p1)
```

ggsem *Launch ggsem Shiny Application*

Description

Main function to launch the ggsem Shiny application for interactive network and structural equation modeling visualization. The app can be started with pre-existing objects or used to create visualizations from scratch.

Usage

```
ggsem(
  object = NULL,
  model_obj = NULL,
  model = NULL,
  metadata = NULL,
  type = "sem",
  session = "sem",
```

```
  center_x = NULL,
  center_y = NULL,
  width = NULL,
  height = NULL,
  random_seed = NULL,
  group_id = NULL,
  group_level = NULL
)
```

Arguments

object	Optional visualization object. Supported types include: - For SEM: 'lavaan', 'qgraph' (from semPaths), 'sem_graph' (tidySEM), 'MxRAMModel' (OpenMx), 'mplusObject', 'grViz' (diagrammeR) - For networks: 'igraph', 'network', 'qgraph'
model_obj	Optional model object to accompany visualization objects. Required for some object types like 'sem_graph' (tidySEM) for SEM visualizations.
model	Same with model_obj
metadata	Path to a saved RDS file containing a previous ggsem workflow session or RDS-read object. When provided, the app will load this session data instead of creating a new one.
type	Type of analysis: 'sem' for structural equation modeling or 'network' for network analysis. Default is 'sem'.
session	Initial session type (element type) when app launches. Either 'point', 'line', 'annotation', 'loop', 'sem' or 'network'. Default is 'sem'.
center_x	X-coordinate for the center of the visualization. If NULL, defaults to 0.
center_y	Y-coordinate for the center of the visualization. If NULL, defaults to 0.
width	Width of the visualization area. If NULL, defaults to 25.
height	Height of the visualization area. If NULL, defaults to 25.
random_seed	Random seed for reproducibility of layouts. If NULL, uses current time in milliseconds.
group_id	Identifier for grouping in multi-group models
group_level	Level for grouping in multi-group models as in original data file or model object

Details

The ggsem Shiny application provides an interactive interface for: - Visualizing and customizing SEM path diagrams - Creating and modifying network visualizations - Adjusting node/edge aesthetics, layouts, and annotations - Exporting high-quality publication-ready graphics

When starting with an object, the app will pre-load the visualization and allow further customization. When starting without an object, users can upload data or use built-in examples.

Value

Launches a Shiny application. Does not return a value.

Supported Object Types

SEM Objects:

- `lavaan`: Fitted lavaan models
- `qgraph`: semPaths objects from lavaan models
- `sem_graph`: tidySEM objects
- `MxRAMModel`: OpenMx models
- `mplusObject`: Mplus models
- `grViz`: diagrammeR objects from lavaanPlot

Network Objects:

- `igraph`: igraph network objects
- `network`: network package objects
- `qgraph`: qgraph network objects

Examples

```
## Not run:
# Launch app without pre-existing objects
ggsem()

# Launch app with a lavaan model
library(lavaan)
model <- ' visual  =~ x1 + x2 + x3
           textual =~ x4 + x5 + x6
           speed   =~ x7 + x8 + x9 '
fit <- cfa(model, data = HolzingerSwineford1939)
ggsem(object = fit)

# Launch app with an igraph network
library(igraph)
g <- make_ring(10)
ggsem(object = g, type = "network", session = "network")

# Launch app with custom dimensions
ggsem(object = g, type = "network", center_x = 10, center_y = 10, width = 30, height = 30)


# Launch app with no input
ggsem()

## End(Not run)
```

ggsem_builder *Create multi-group SEM visualizations*

Description

Build and launch multi-group SEM visualizations with an intuitive API. Ideal for comparing multiple groups from the same or different models.

Usage

```
ggsem_builder(type = "sem")
```

Arguments

type	Default type for all groups: 'sem' or 'network' (default: 'sem')

Value

A ggsem_builder object

ggsem_launch *Launch ggsem Shiny application*

Description

Launches the ggsem Shiny application for interactive SEM and network visualization.

Usage

```
ggsem_launch(x = NULL, ...)
```

Arguments

x	A ggsem_builder object, model object, visualization object, or NULL
...	Additional arguments passed to Shiny app

Value

Runs Shiny application

ggsem_launch.default *Launch ggsem Shiny application from objects*

Description

Launch ggsem Shiny application from objects

Usage

```
## Default S3 method:
ggsem_launch(x, center_x = NULL, center_y = NULL, ...)
```

Arguments

x	Any compatible R object (lavaan, igraph, qgraph, etc.)
center_x	X-coordinate for center position
center_y	Y-coordinate for center position
...	Additional arguments passed to Shiny app

Value

Runs Shiny application

ggsem_launch.ggsem_builder *Launch ggsem Shiny application from builder*

Description

Launch ggsem Shiny application from builder

Usage

```
## S3 method for class 'ggsem_builder'
ggsem_launch(x, session = NULL, ...)
```

Arguments

x	A ggsem_builder object
session	Session type ('sem' or 'network')
...	Additional arguments passed to Shiny app

Value

Runs Shiny application

ggsem_silent *Run ggsem silently (without launching the app) and get the visualization outputs*

Description

This function processes a saved ggsem workflow metadata file and extracts all visualization data (points, lines, annotations, loops) that would be displayed in the Shiny app. It reproduces both SEM and network visualizations from the saved session state.

Usage

```
ggsem_silent(metadata)
```

Arguments

metadata	A file path of metadata or a list containing ggsem workflow metadata, typically loaded from an RDS file saved by the ggsem Shiny app using the "Export Workflow" functionality. The metadata should contain SEM groups, network groups, visual elements, and group labels.

Value

A list of four data frames:

- points - Data frame containing point coordinates and properties
- lines - Data frame containing line coordinates and properties
- annotations - Data frame containing text annotations
- loops - Data frame containing loop coordinates and properties

Examples

```
## Not run:
# Load a saved ggsem workflow
workflow_metadata <- readRDS("ggsem_workflow_20240101_120000.rds")

# Extract visualization data
viz_data <- ggsem_silent(workflow_metadata)

# Access the different components
points <- viz_data$points
lines <- viz_data$lines
annotations <- viz_data$annotations
loops <- viz_data$loops

## End(Not run)
```

launch *Launch method for ggsem_builder objects*

Description

Convenience method for builder pattern: builder |> launch()

Usage

```
launch(builder, ...)
```

Arguments

builder	A ggsem_builder object
...	Additional arguments passed to Shiny app

Value

Runs Shiny application

metadata_to_ggplot *Convert ggsem workflow metadata directly to a ggplot object*

Description

This function is a convenient wrapper that takes ggsem workflow metadata, extracts the visualization data silently, and converts it to a ggplot object in one step.

Usage

```
metadata_to_ggplot(
  metadata,
  element_order = c("lines", "points", "loops", "annotations"),
  zoom_level = 1,
  horizontal_position = 0,
  vertical_position = 0,
  n = 100
)
```

Arguments

`metadata`	A list containing ggsem workflow metadata, typically loaded from an RDS file saved by the ggsem Shiny app using the "Export Workflow" functionality or the files directory.
`element_order`	A character vector specifying the order in which graphical elements are added to the plot. For example: `c("annotations", "loops", "lines", "points")`. Elements at the front appear on top. Default includes all elements.
`zoom_level`	A numeric value controlling the zoom level of the plot. A value >1 zooms in; <1 zooms out. Default is `1`.
`horizontal_position`	A numeric value to shift the plot horizontally. Default is `0`.
`vertical_position`	A numeric value to shift the plot vertically. Default is `0`.
`n`	Number of points used for interpolation in gradient or curved lines. Default is `100`.

Details

This function combines the functionality of `ggsem_silent` and `csv_to_ggplot` into a single convenient call. It's useful when you want to go directly from saved workflow metadata to a ggplot object without intermediate steps.

Value

A ggplot object with an `axis_ranges` attribute specifying the x and y axis ranges after adjustments.

Examples

```
## Not run:
# Load a saved ggsem workflow
workflow_metadata <- readRDS("ggsem_workflow_metadata.rds")

# Convert directly to ggplot
p <- metadata_to_ggplot(
  metadata = workflow_metadata
)

# Customize the plot further
p + ggtitle("My SEM Visualization")

## End(Not run)
```

save_figure *Save a ggplot Object with Adjusted Dimensions*

Description

This function saves a ggplot object (created from 'csv_to_ggplot()' function) to a file with dimensions automatically determined based on the x-axis and y-axis ranges of the plot. The size of the output can be further controlled using addtional arguments.

Usage

```
save_figure(
  filename,
  plot,
  units = "in",
  dpi = 300,
  aspect_ratio = NULL,
  scale_factor = 0.122,
  zoom_level = 1,
  ...
)
```

Arguments

filename	A string. The name of the output file (e.g., "plot.png").
plot	A ggplot object to save.
units	A string. Units for width and height. Default is "in" (inches). Other options include "cm" or "mm".
dpi	Numeric. Resolution of the output file in dots per inch. Default is 300.
aspect_ratio	Numeric or NULL. If provided, fixes the aspect ratio of the plot (e.g., 1 for square). If NULL, uses the natural data aspect ratio. Default is NULL.
scale_factor	Numeric. A scaling factor to control the overall size of the saved plot. Default is 0.122.
zoom_level	Numeric. A zoom factor, value > 1 means zoom out. Default is 1.0.
...	Additional arguments passed to ggsave().

Value

Saves the ggplot object to the specified file and does not return a value.

Examples

```
## Not run:
# CSV files from ggsem app
points_data <- data.frame(
```

```
x = 20, y = 20, shape = 'rectangle', color = '#D0C5ED', size = 50,
border_color = '#9646D4', border_width = 2, alpha = 1,
width_height_ratio = 1.6, orientation = 45, lavaan = FALSE,
network = FALSE, locked = FALSE, group = 1
)

lines_data <- data.frame(
x_start = 2, y_start = -2, x_end = 10, y_end = -2, ctrl_x = NA, ctrl_y = NA,
ctrl_x2 = NA, ctrl_y2 = NA, curvature_magnitude = NA, rotate_curvature = NA,
curvature_asymmetry = NA, type = 'Straight Line', color = '#000000',
end_color = NA, color_type = 'Single',
gradient_position = NA, width = 1.5, alpha = 1, arrow = FALSE,
arrow_type = NA, arrow_size = NA, two_way = FALSE, lavaan = FALSE,
network = FALSE, line_style = 'solid', locked = FALSE, group = 1
)

p1 <- csv_to_ggplot(graphics_data = list(points_data, lines_data),
              zoom_level = 1.2, # Value from the ggsem app
              horizontal_position = 0, # Value from the ggsem app
              element_order = c('lines', 'points')) # order priority: lines < points

# Save with default scaling
save_figure("p1.png", p1)

## End(Not run)
```

`set_type` *Set default type for the builder*

Description

Change the default type for subsequent add_group() calls

Usage

```
set_type(builder, type = "sem")
```

Arguments

`builder`	A ggsem_builder object
`type`	Default type: 'sem' or 'network'

Value

Updated ggsem_builder object

Index